Amendments to the Claims

Listing of the Claims

1. (Currently amended) A tool for processing a p-code file, comprising:

analyzing p-code methods to be compiled within said p-code file;

identifying ~~those~~ one or more p-code methods ~~to be compiled within the file having associated with them~~ that have at least one profile parameter including an associated priority level above a threshold level; and

annotating said identified p-code methods to be compiled ~~in a manner adapted~~ to enable preferential processing of said p-code file based on said associated priority level of each identified p-code methods~~ to be compiled by a compiler~~.

2. Cancelled

3. (Currently amended) The tool of claim 1, wherein:

said p-code file comprises ~~a Java~~ an application file ~~including Java classes, said Java class being annotated in a manner adapted to enable preferential~~ for processing ~~of said identified Java classes~~ by a ~~Java~~ virtual machine (VM) just-in-time (JIT) compiler.

4. (Original) The tool of claim 1, wherein said annotations are provided in-line with said identified p-code methods.

5. (Original) The tool of claim 1, wherein said annotations are provided as a separate file.

6. (Original) The tool of claim 1, wherein:

said at least one profile parameter comprises at least one of a method execution time, a frequency of method invocation, a number of instructions and a use of loop structures.

7. (Original) The tool of claim 1, wherein:

said at least one profile parameter comprises at least one of an execution time parameter, an input/output utilization parameter and a processor utilization parameter.

8. (Original) The tool of claim 1, wherein: said analyzing comprises identifying at least one of a static profile parameter and a dynamic profile parameter.

9. (Currently amended) The tool of claim [[2]] 1, wherein:

said annotation comprises setting a normally unused bit within a method access flag field of an identified Java class file.

10. (Currently amended) The tool of claim [[2]] 1, wherein:

said annotation comprises selectively setting each of a plurality of normally unused bits within a method access flag field of an identified ~~Java~~ class file, wherein said unused bits are selectively set to define thereby a priority level of a respective annotated method.

11. (Currently amended) The tool of claim 3, wherein:

each identified byte-code portion of said ~~Java~~ application is associated with one of a plurality of priority levels, said annotation being indicative of respective priority levels.

12. (Previously presented) The tool of claim 3, further comprising:

selectively pre-compiling at least a portion of said application file.

13. (Original) The tool of claim 12, wherein: said precompiled portion of said application file is included within a virtual machine.

14. Cancelled.

15. Cancelled.

16. (Currently amended) A method of adapting the interpretation of a p-code method within a p-code file by a virtual machine (VM), comprising:

identifying one or more p-code methods that have at least one profile parameter including an associated priority level above a threshold level;

compiling p-code methods within a p-code file in a ~~priority~~ prioritized manner associated with compilation priority indicative annotation; and

storing said compiled p-code methods in a cache for subsequent execution in place of corresponding interpreted p-code methods.

17. Cancelled.

18. (Currently amended) The method of claim 16, wherein:

said p-code file comprises ~~a Java~~ an application file ~~including Java classes and annotated Java classes, said annotated Java classes being preferentially compiled~~ for processing by a ~~Java~~ virtual machine (VM) just-in-time (JIT) compiler.

19. (Original) The method of claim 16, wherein said annotations are provided in-line with said identified p-code methods.

20. (Original) The method of claim 16, wherein said annotations are provided as a separate file.

21. (Original) The method of claim 16, further comprising:

in response to cache memory utilization above a threshold level, prioritizing the contents of said cache memory.

22. (Original) The method of claim 21, wherein:

said cache memory contents are prioritized by deleting from said cache compiled code associated with a least recently executed method.

23. (Original) The method of claim 21, wherein:

said cache memory contents are prioritized by deleting from said cache compiled code associated with a previously compiled method having a lower priority level than a presently compiled method.

24. (Original) The method of claim 20, wherein:

compiled byte-code stored in said cache is accessed via a cache map, said cache map being updated in response to a change in cache utilization.

25. (Currently amended) The method of claim 18, further comprising:

compiling non-annotated byte-code within said ~~Java~~ application if said non-annotated byte-code utilizes VM resources beyond a threshold level.

26. (Original) The method of claim 25, wherein:

said compiled non-annotated byte-code is assigned a priority level in accordance with said utilized VM resources.

27. (Original) The method of claim 26, wherein:

said priority level of said annotated byte-code is further adapted in accordance with said utilized VM resources.

28. (Original) The method of claim 20, further comprising:

said compiled annotated byte-code is assigned a priority level in accordance with said utilized VM resources.

29. (Original) The method of claim 28, wherein:

said priority level of said annotated byte-code is further adapted in accordance with said utilized VM resources.

30. (Original) The method of claim 26, wherein said VM resources comprise at least one of an execution time parameter, an input/output utilization parameter and a processor utilization parameter.

31. (New) The method of claim 1 wherein said at least one profile parameter is stored as a method attribute in an attribute table.

32. (New) The method of claim 16 wherein said at least one profile parameter is stored as a method attribute in an attribute table.